SciDAC
Scientific Discovery through Advanced Computing

Performance Sensitivity of HPC Applications

Allan Snavely, SDSC

Workshop on Performance Characterization, Modeling, and Benchmarking for HPC Systems

Emeryville, May 6th 2003

Outline

- Some philosophy about performance modeling
  - What *is* it exactly and how can it be useful?
- An update on the state-of-performance modeling
  - Where do we (particularly PERC and collaborators including HPCMP stand today and where are we going?)
- Some results – performance sensitivity of HPC applications

What is a performance model?

- A performance model is not…(a lot of things that get confounded with one)
- A performance model is…
  - A calculable explanation of why a {program, application,input} tuple performs as it does
- Should yield a prediction (quantifiable objective)
- Performance models embody *understanding* of the factors that affect performance
  - Inform the tuning process (of application and machine)
  - Guide applications to the best machine
  - Enable applications driven architecture design
  - Extrapolate to the performance of future systems

Goals for performance modeling tools and methods

- Generation of performance models should be automated, or at least as regular and systemized as possible
- Performance models *must* be time-tractable
- Error is acceptable if it is bounded and allows meeting these objectives

A Road Map to Ubiquitous Performance Modeling

- Research to develop methods
  - – Results exhibited via “proof-of-principle”

We are here

- Hardening
  - – Modeling studies of full HPC applications
- Tool-making to render modeling ubiquitous
  - – Overhead of tools must be light and recipe for using them very clear

Where Modeling Fits In

- The HPCMP procurement process
  - Identification of strategic applications (program level)
  - Extraction of typical calculations (benchmarking team)
  - Gathering application signatures of typical calculations (PMaC, benchmarking team, end-users)
  - Profiling candidate machines (PMaC, Instrumental)
  - Modeling
  - Assigning weights (Cray, Larry) and solving an optimization problem (Bill Ward)

A useful framework

- **Machine Profiles** - characterizations of the rates at which a machine can (or is projected to) carry out fundamental operations abstract from the particular application.
- **Application Signature** - detailed summaries of the fundamental operations to be carried out by the application independent of any particular machine.

Combine Machine Profile and Application Signature using:

- **Convolution Methods** - algebraic mappings of the Application Signatures on to the Machine profiles to arrive at a performance prediction.

Developing the Framework

- Work in the first year was focused primarily on instantiating the framework
  - Results were cast primarily in terms of predication accuracy (to validate models) and were applied first to kernels (ie. PETSc, NPB, TableToy), then mini and synthetic apps. (i.e synNLOM) then full applications (i.e. POP, Cobalt)
  - Along the way research was advanced in reducing tracing time, improving granularity of models, and validating the underlying convolutions

Technology improvements

to render full HPC applications modeling tractable

Porting to DynINST API enabled low-overhead tracing via sampling

Practically, we have found techniques for generating approximated traces via sampling can reduce tracing time while preserving reasonable trace fidelity [1]. Also we found that representing traces by a dynamic CFG decorated with instructions (especially memory instructions) characterized by memory access pattern can reduce the size of stored trace files by three orders of magnitude [2]. These improvements in the space and time required for tracing have now rendered full-application modeling tractable. In some cases it is possible to obtain reasonably accurate traces and resulting performance models from 10% or even 1% sampling with little slowdown of the instrumented program vs. un-instrumented execution.

1. L. Carrington, A. Snavely, N. Wolter, X. Gao, "A Performance Prediction Framework for Scientific Applications", *Workshop on Performance Modeling and Analysis* - *ICCS*, Melbourne, June 2003

2. X. Gao, A. Snavely, "Exploiting Stability to Reduce Time-Space Cost for Memory Tracing", *Workshop on Performance Modeling and Analysis* - *ICCS*, Melbourne, June 2003

Papers available at www.sdsc.edu/PMaC

POP
Parallel Ocean Program

POP has been ported to a wide variety of computers for eddy-resolving simulations of the world oceans and for climate simulations as the ocean component of coupled climate models.

Table 2: **Real versus Predicted-by-Model Wall-clock Times for POP x1 Benchmark**
X1 at 16 processors real time 9.21 seconds, predicted time 9.79 seconds, error 6.3%

# of pe’s	**Blue Horizon**			**Lemieux**		
	Real Time(sec)	**Predicted Time(sec)**	**Error**	**Real Time(sec)**	**Predicted Time(sec)**	**Error**
16	204.92	214.29	-5 %	125.35	125.75	0 %
32	115.23	118.25	-3 %	64.02	71.49	-11 %
64	62.64	63.03	1 %	35.04	36.55	-4 %
128	46.77	40.60	13 %	22.76	20.35	11 %

# of pe’s	**Longhorn**			**Seaborg**		
	Real Time(sec)	**Predicted Time(sec)**	**Error**	**Real Time(sec)**	**Predicted Time(sec)**	**Error**
16	93.94	95.15	-1 %	204.3	200.07	2 %
32	51.38	53.30	-4%	108.16	123.10	-14%
64	27.46	24.45	11%	54.07	63.19	-17%
128	19.65	15.99	16%%	45.27	42.35	6 %

POP

- Seconds per simulation day

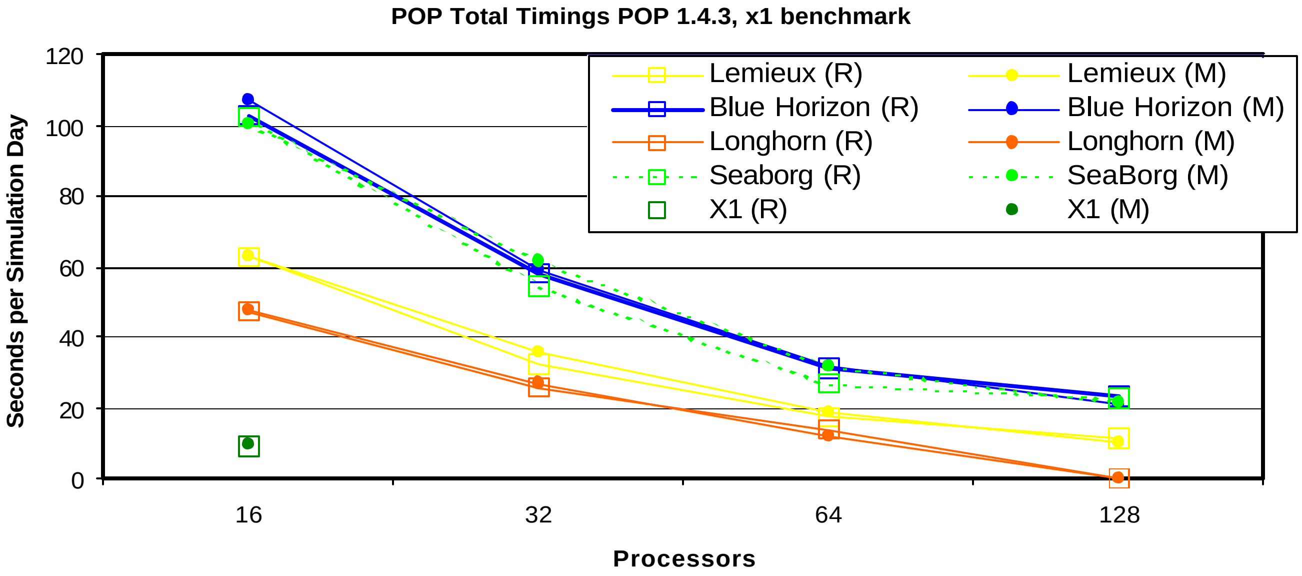

Figure 2: Real (R) versus Predicted-by Model (M) Times for POP x1 Benchmark

Explaining Relative Performance of POP

?? In Case 1, we model the effect of reducing the bandwidth of BH's network to that of a single rail of the Quadrics switch. There is no discernable performance effect as the POP x1 problem at this size is not sensitive to a change in peak network bandwidth from 350MB/s to 269 MBs.

?? In Case 2 we model the effect of replacing the Colony switch with the Quadrics switch. There is a significant performance improvement due to the 5 ?s latency of the Quadrics switch versus the 19 ?s latency of the Colony switch. This is because the barotropic calculations in POP x1 at this size are latency sensitive.

?? In Case 3 we use Quadrics latency but Colony bandwidth just for completeness.

?? In Case 4 we model keeping the Colony switch latencies and bandwidths but replacing the Pwr3 processors and local memory subsystem with Alpha ES640 processors and their memory subsystem. There is a substantial improvement in performance due mainly to the faster memory subsystem of the Alpha. The Alpha can load stride-1 data from its L2 cache at about twice the rate of the Pwr3 and this benefits POP x1 a lot.

?? The last set of bars show the values of TCS performance, processor and memory subsystem speed, network bandwidth and latency, as a ratio to BH's values.

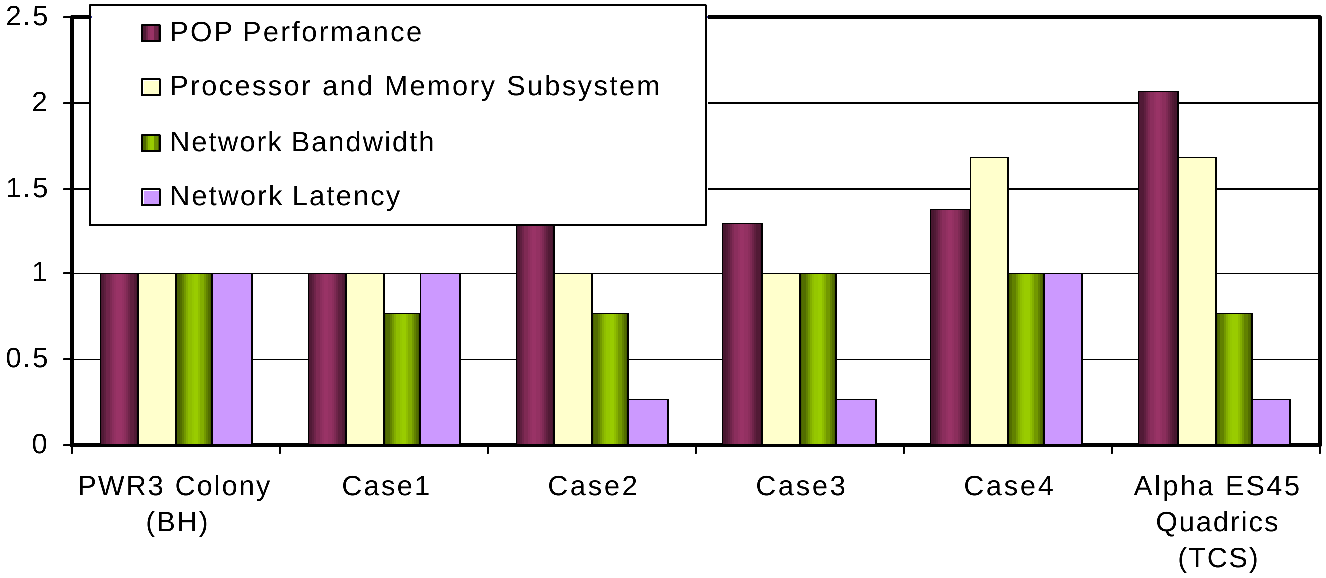

Figure 3: Modeled Contributions to Lemeuix's (TSC) performance improvement over Blue Horizon on POP x1 at 16 CPUs

POP Performance Sensitivity

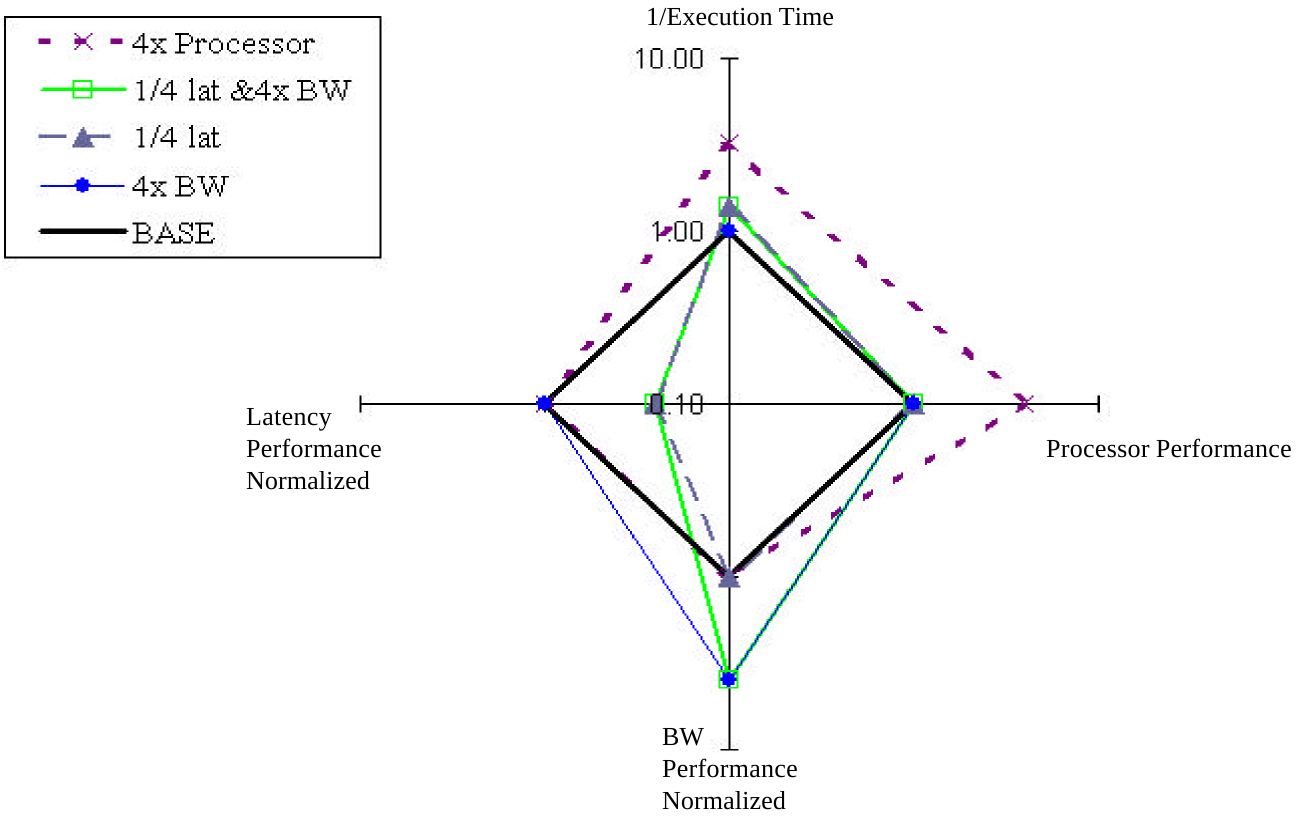

Figure 4: POP Performance Sensitivity for 128 cpu POP x1. Axis are plotted logscale and normalized to 1, thus the black quadrilateral represents the {execution time, network bandwidth, network bandwidth, CPU and memory subsystem performance} of BH.

Navy Layered Ocean Model

- The Navy's hydrodynamic non-linear primitive equation layered ocean circulation model has been used at NOARL for more than 10 years for simulations of the ocean circulation in the Gulf of Mexico, Carribean , Pacific, Atlantic, and other seas and oceans.

Table 3: Predictions of 28 CPU synNLOM for different machines

Machine	Real Time (s)	Predicted Time (s)	% Error
PSC's Lemieux	1818	1816	0.1
SDSC's Blue Horizon	4462	4594	-3.0
NERSC's Seaborg	4375	4756	-8.7
TACC's Longhorn	1944	1872	3.7

The results of Table 3 were obtained using 1% sampling. We estimate a full trace would take more than a month to obtain on 28 processors! NLOM is reasonably regular and the low error percentages in Table 3 do not seem to justify doing a full trace although the code is important enough to DoD that they would provide a month of system time for the purpose.

synLOM zoom-in on node

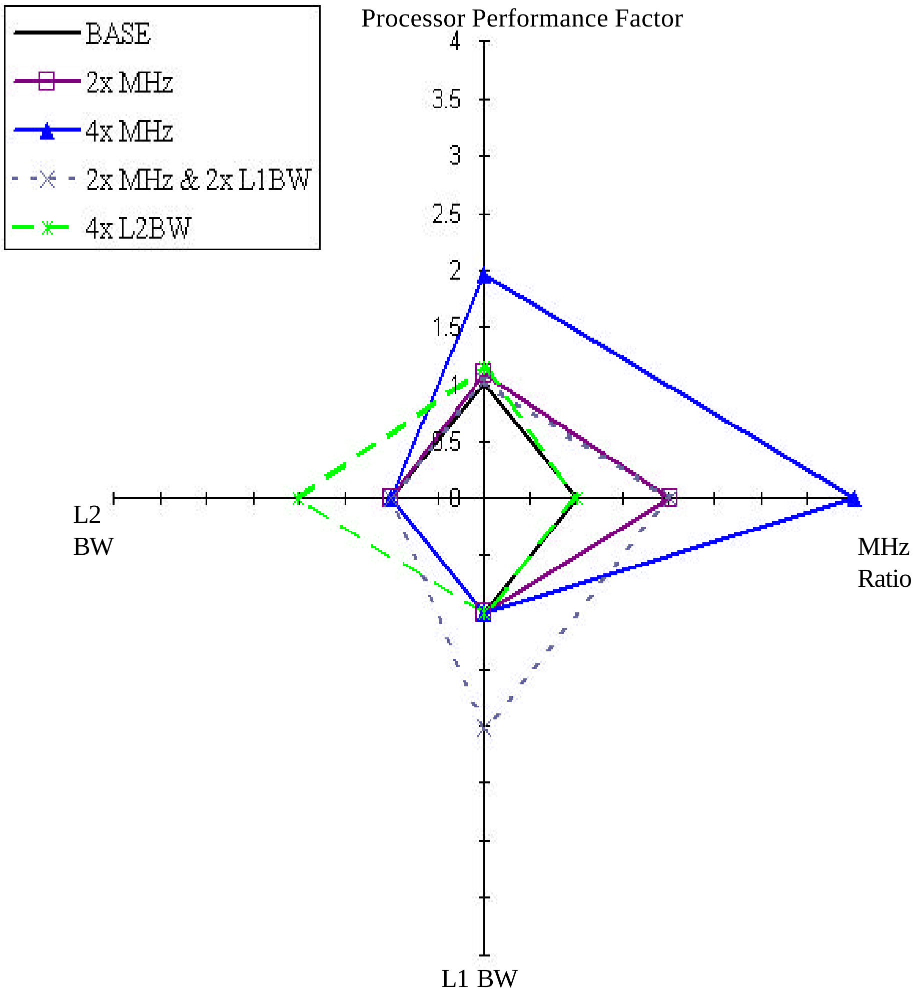

Figure 5: Breakdown of effect on processor components for synLOM 28 CPU

Cobalt

Cobalt60 is an unstructured Euler/Navier-Stokes flow solver that is routinely used to provide quick, accurate aerodynamic solutions to complex CFD problems. Cobalt60 handles arbitrary cell types as well as hybrid grids that give the user added flexibility in their design environment. It is a robust HPC application that solves the compressible Navier-Stokes equations using an unstructured Navier-Stokes solver. It uses Detached-Eddy Simulation (DES) which is a combination of Reynolds-averaged Navier-Stokes(RANS) models and Large Eddy Simulation (LES).

We ran 7 iterations of a tunnel model of an aircraft wing with a flap and endplates with 2,976,066 cells that runs for about ½ hour on 4 CPUs of BH. We used a 2-step trace method to ascertain in the first phase that 70% of the time is spent in just one basic block. We then applied 1% sampling to this basic block and 10% sampling to all the others in the second step of MetaSim tracing. At time of writing we have only verified models for PSC Lemeiux at 4, 32, 64, and 128 CPUS with average of less than 5% error.

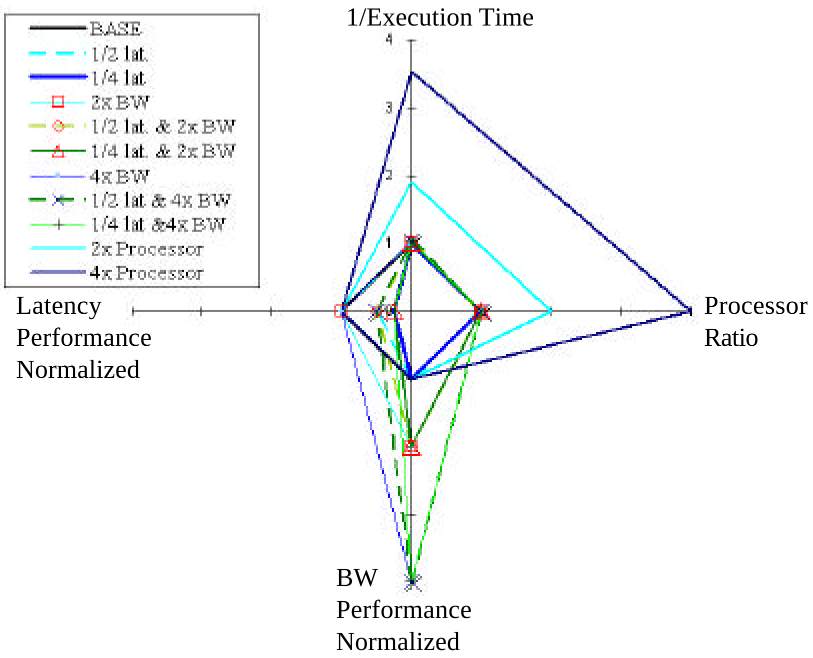

Figure 6: **Cobalt 32 CPU Sensitivity Study**

Cobalt

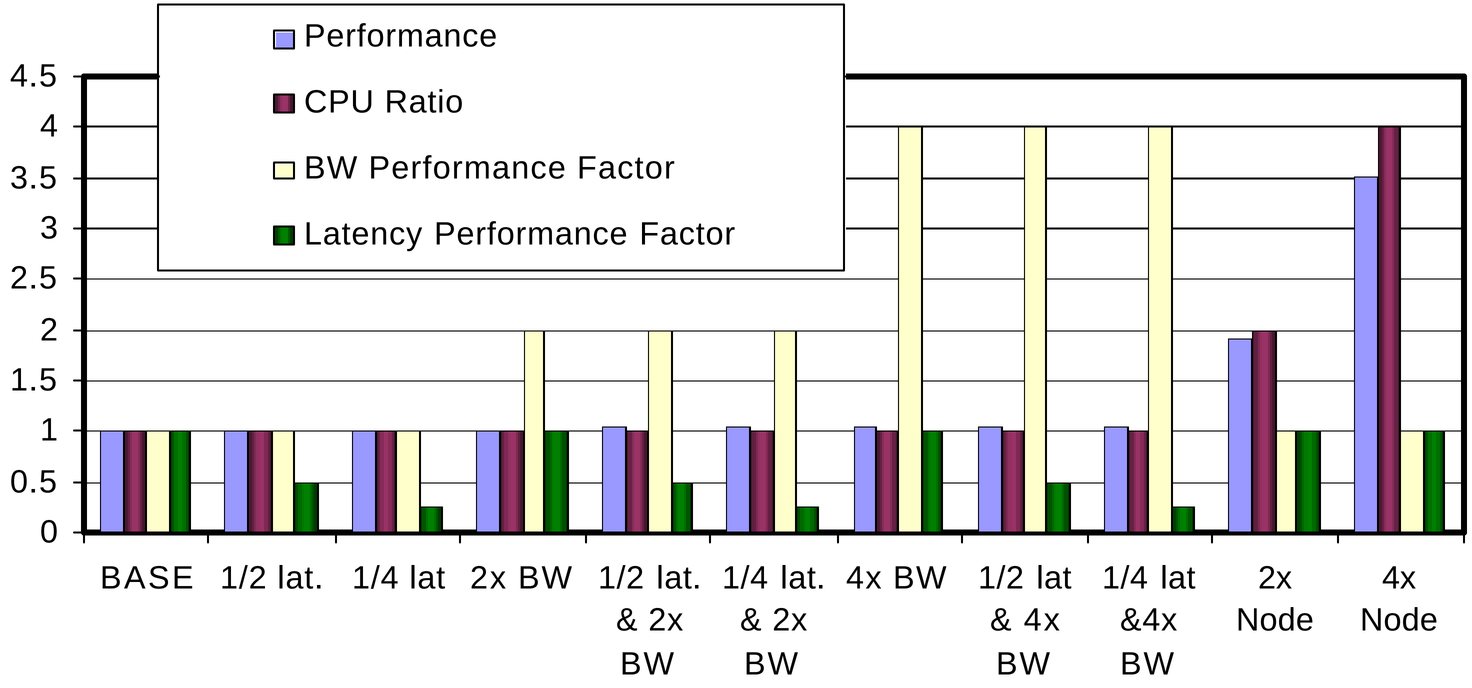

Figure 7: **Cobalt 32 CPU Sensitivity Study**

Workload characterization

Larger CPU count POP x1 problems become more network latency sensitive and remain not-very bandwidth sensitive.

synNLOM is somewhat more network bandwidth sensitive than POP because it sends less frequent, larger messages.

Cobalt60 is most sensitive to improvements in the processor performance at this size and this remains true at larger processor counts.

As a trivial example, if one’s workload included more synLOM and less POP, one would be willing to spend more $ to improve network bandwidth.

Conclusion

A systematic method for generating performance models of HPC applications has advanced via efforts of this team members and others in the performance modeling community and has begun to make performance modeling systematic, time-tractable, and thus generally useful for performance investigations. It is reasonable now to make procurement decisions based on the computational demands of the target workload.

It is reasonable now to tune current systems and influence the implementation of near-future systems informed by the computational demands of the target workload. Team members have recently used these methods to tune the production system IBM Blue Horizon at SDSC and to inform the design of IBM BG/L and begun to look at IBM Blue Planet. It is reasonable now to design future systems based on the quantified performance implications of hardware features for characterized workloads. Members of the team are participating in the DARPA HPCS program to provide workload characterization and modeling towards this goal.

The goal of ubiquitous performance modeling; to widely disseminate tools and methods for modeling is *not* yet realized but we think it can and should be.